

Mini Manual de Usuario de Repose

Maximiliano Cristiá
mcristia@flowgate.net
Flowgate Security Consulting
Rosario – Argentina

Abril de 2009

Resumen

Este es un manual de usuario muy breve para Repose. Repose es una herramienta para obtener y validar requerimientos funcionales, basada en prototipación rápida desechable y software de oficina abierto y gratuito.

Índice

1. Obtención y Validación de Requerimientos Funcionales	1
1.1. Metodología para la obtención y validación de requerimientos funcionales basada en prototipación rápida desechable	2
1.1.1. Desarrollo de prototipos	3
1.1.2. Validación cruzada	3
1.1.3. Ejemplos: la forma de definir prototipos	5
1.1.4. Los prototipos son desechables... pero no se desechan	5
1.2. Repose: una herramienta para prototipación rápida desechable	5
1.2.1. Requisitos para instalar la herramienta	6
1.2.2. Panorama general de Repose	6
1.2.3. Cómo usar Repose	7
1.2.4. Compilar, ejecutar y modificar el prototipo	17

1. Obtención y Validación de Requerimientos Funcionales

Las tres cosas más importantes para obtener y validar requerimientos funcionales son:

1. Nunca ir a una reunión con un cliente sin un prototipo.
2. Nunca ir a una reunión con un cliente sin un prototipo.
3. Nunca ir a una reunión con un cliente sin un prototipo.

Es tan evidente el acierto de esta frase como lo es la necesidad de definir una metodología y herramientas para llevarla a la práctica. ¿Qué es un prototipo? ¿Cuánto tiempo lleva desarrollarlo? Si hablamos de prototipos desechables, ¿cuánto vamos a invertir en desarrollarlos si luego hay que tirarlos? ¿Cuándo deja de ser beneficioso desarrollar un prototipo para obtener el requerimiento correcto y comienza a ser más rentable implementar el programa incorrecto para luego corregirlo? Si

hay que ir a cada reunión con el cliente con un prototipo, ¿cuánto podemos demorar en desarrollar un prototipo sin alargar de forma absurda la ingeniería de requerimientos? Si desarrollar un prototipo implica programar, ¿quién debe desarrollarlos? ¿Los ingenieros de requerimientos a quienes les pedimos ser menos técnicos, conocer más el dominio de aplicación y describir los requerimientos sin hablar en términos de implementación? ¿Los programadores, entonces, quienes no estuvieron con el cliente y a los cuales hay que transmitirle lo que el cliente dijo? ¿No podríamos introducir más errores al comunicarles los *supuestos* requerimientos?

Claramente, para cumplir con el principio postulado por Schrage el desarrollo de prototipos debe verificar las siguientes condiciones:

1. Desarrollar un prototipo para dos o tres requerimientos no debe consumir más de dos o tres horas.
2. Modificar un prototipo para adaptarlo a las correcciones indicadas por el cliente durante la validación del mismo, debe demorar menos de quince minutos.
3. Desarrollar un prototipo no debería implicar programar, pues así los ingenieros de requerimientos sin demasiados conocimientos técnicos podrán hacerlo.

En las secciones siguientes se presenta una metodología para obtener y validar requerimientos funcionales utilizando casi exclusivamente prototipos desechables (1.1) y una herramienta que da soporte o asistencia a esa metodología (1.2).

Ahora, ¿por qué prototipos desechables y no prototipos *evolutivos*? La razón es muy sencilla: el objetivo de los prototipos evolutivos es entregar un sistema en funcionamiento al cliente, en tanto que el objetivo de los prototipos desechables es obtener y/o validar los requerimientos [Som95]. Pero, ¿por qué no ir desarrollando un sistema a partir de uno que implemente, mal o bien, los primeros requerimientos como propone la comunidad de la prototipación evolutiva? Porque en general no será posible diseñar correctamente un sistema desarrollado de esa forma, y un buen diseño (es decir un diseño que pueda incorporar eficientemente los cambios más probables) es la clave para un sistema flexible y duradero. Trataremos extensivamente el diseño de software en capítulos posteriores.

1.1. Metodología para la obtención y validación de requerimientos funcionales basada en prototipación rápida desechable

La metodología que proponemos, graficada en la Figura 1, comienza con una primera reunión con los referentes clave del sistema (gerentes, directores, jefes de área, socios, etc.) en la cual, a requisitoria de los ingenieros, deben presentar el negocio detrás del sistema y una exposición general sobre los requerimientos. El contenido y estructura de la reunión deben ser comunicados a los referentes con suficiente antelación, pues de lo contrario esta primera reunión seguramente será un fracaso. Si el cliente logra presentar satisfactoriamente el negocio detrás del sistema, se pasa a la etapa siguiente; caso contrario, se debe asistir al cliente en elaborar el negocio detrás del sistema.

El segundo paso de nuestra metodología consiste en listar todos los interesados relevantes, teniendo en cuenta el contenido de la etapa previa. Esta tarea puede, y de hecho ocurre muy frecuentemente, no completarse en un solo paso por lo que la lista se va completando a medida que avanza la ingeniería de requerimientos. Para determinar, al menos, una lista preliminar se debe considerar el dominio de aplicación según surge de la exposición del negocio detrás del sistema y de los requerimientos mencionados por el cliente en la etapa anterior, y proyectarlos sobre el organigrama de la empresa, sus proveedores, clientes, asociados, organismos de control, etc. Por ejemplo, si el negocio detrás del sistema dice algo como “con este nuevo sistema de control de *stock* se espera que la empresa reduzca en un 50 % los retrasos en las entregas lo que redundará en mayor satisfacción de los clientes y un consecuente aumento en las ventas”, entonces los primeros interesados serán: el área de almacenes,

logística, compras, proveedores, atención al cliente, etc. Probablemente con el correr del proyecto algunos de estos primeros interesados dejarán de serlo y aparecerán otros. Por otro lado, se puede recurrir a alguna lista de referencia de potenciales interesados y utilizarla como *checklist*. La lista inicial deberá incluir a todos los participantes de la primera reunión.

1.1.1. Desarrollo de prototipos

Luego de la primera reunión con el cliente los ingenieros deberían construir un primer prototipo para llevarlo a la siguiente reunión con los interesados, como se explica en la sección 1.2. La segunda reunión y las subsiguientes deben estructurarse alrededor del uso y/o modificación de los prototipos que el equipo de ingenieros vayan desarrollando. A las primeras reuniones deben asistir los interesados que parezcan ser los más prometedores para darnos los requerimientos fundamentales del sistema. En cada una de estas reuniones, que seguramente durarán no más de un par de horas, los ingenieros alentarán a los interesados a utilizar el prototipo y a plantear sus diferencias y expectativas sobre el sistema final. Preguntarán a los interesados, a partir del negocio detrás del sistema y de los requerimientos generales con que ya cuentan, sobre requerimientos más específicos. Uno, dos o tres requerimientos específicos son suficientes como para dar por finalizada la reunión. Por ejemplo, requerimientos más específicos pueden ser “la planta nos pasa un pedido de materiales, nosotros los buscamos en el almacén, les entregamos los que podemos y los damos de baja en el sistema, hacemos un pedido a compras de los que no tenemos disponibles, y nos sentamos a esperar a que lleguen”. Al finalizar la reunión se acuerda una fecha, hora, lugar y participantes para la siguiente.

A menos que las reuniones sean especialmente productivas, desarrollar los primeros prototipos no debería consumir más de entre 2 y 10 horas-hombre (los primeros llevarán cerca de 10 horas-hombre en tanto que los siguientes requerirán menos tiempo). De esta forma podría pactarse una reunión cada 48 horas.

El equipo de ingenieros asiste a cada reunión con un prototipo. Los interesados “juegan” con el prototipo guiados por los ingenieros, quienes les presentan una serie de ejemplos de cómo usar el sistema para observar los resultados que este arroja. En el mejor de los casos cada interesado que participa de la reunión debe ejecutar todos y cada uno de los ejemplos y dar su opinión sobre cada uno de ellos. Estas opiniones tomarán la forma de aprobaciones, correcciones importantes o secundarias, indicaciones de incompletitud, etc. Los ingenieros deben tomar notas de todas ellas para corregir y ampliar el prototipo. Algunos de los cambios pueden hacerse durante la misma reunión, como se verá en la sección 1.2.

Es sumamente importante para nosotros notar que durante cada reunión se *obtienen y validan* requerimientos, y se abre el camino para el testing de aceptación, como veremos en los últimos capítulos.

Otro punto a tener en cuenta es que en muchas ocasiones los interesados se dan cuenta de un error, omisión o modificación en los requerimientos varios días después de la reunión y traen el tema luego de que se ha llegado a un acuerdo sobre este. Es parte de la habilidad de los ingenieros resolver estas cuestiones para lograr cumplir con los plazos, presupuesto y parámetros de calidad.

1.1.2. Validación cruzada

En la Figura 2 se muestra con más detalle cómo debería estructurarse cada reunión, aunque los detalles precisos dependen de cosas como si todos los interesados están disponibles en el mismo momento y en el mismo lugar, cuánto tiempo hay para que los interesados estén a disposición de los ingenieros, etc. El punto clave que intentamos resaltar aquí es hacer una *validación cruzada* de cada requerimiento o prototipo. Por validación cruzada entendemos que el prototipo es validado por todos los interesados, si es posible en la misma reunión, de forma tal que ellos mismos resuelvan sus diferencias, intereses, prioridades y expectativas específicas sobre el sistema.

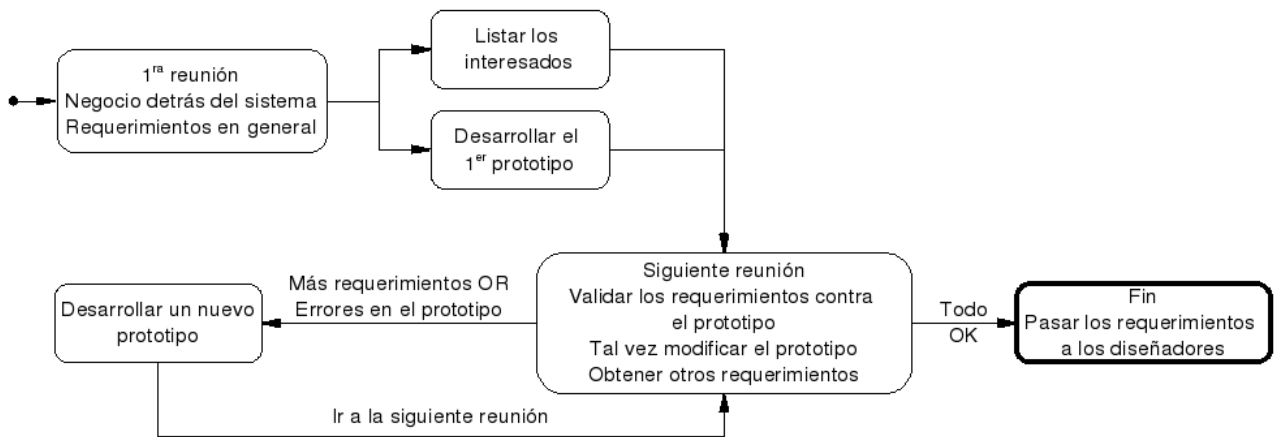


Figura 1: Una descripción general de la metodología.

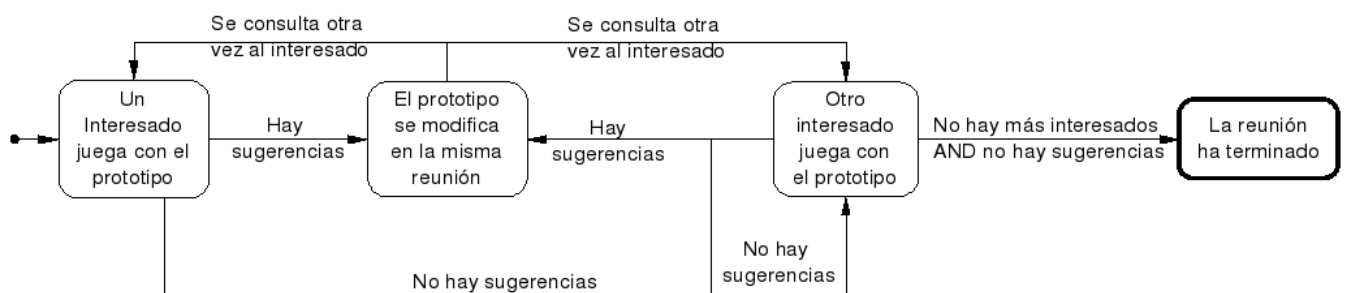


Figura 2: Validar los requerimientos con cada interesado.

La figura sugiere una validación cruzada lineal pero podrían implementarse esquemas más seguros. Por ejemplo, si k interesados han validado un prototipo sin sugerir cambios pero el interesado $k + 1$ hace sugerencias, lo que implica modificar el prototipo, sería conveniente hacer que los k primeros interesados vuelvan a validarlo.

1.1.3. Ejemplos: la forma de definir prototipos

En el contexto de nuestra metodología validar el prototipo o el requerimiento (que en definitiva es la misma cosa), significa que los usuarios “jueguen” o usen el prototipo casi como si fuera el sistema final. Usar o jugar con el prototipo significa, a su vez, ejecutar lo que nosotros denominamos *ejemplos*. Un ejemplo es una forma específica de usar el sistema que define valores concretos para cada una de las variables de entrada, de estado y de salida del sistema o de una parte de este. Al ejecutar un ejemplo el usuario puede observar la salida producida a consecuencia de las entradas que él mismo ingresó. La diferencia sustancial con el sistema terminado es que el prototipo solo puede ejecutar algunos pocos ejemplos. Estos ejemplos son definidos por los ingenieros al construir el prototipo, a raíz de la interacción con los interesados en la reunión actual o en las anteriores. A partir de los ejemplos y de lo que dicen los interesados, los ingenieros enuncian los requerimientos en lenguaje natural de manera tal que cada ejemplo tiene que ser, precisamente, un ejemplo de ese enunciado más general. De esta forma, el prototipo, el enunciado escrito del requerimiento y los ejemplos se sustentan mutuamente para dar mayor solidez, claridad y precisión al pedido de los interesados.

1.1.4. Los prototipos son desechables... pero no se desechan

Los prototipos solo se utilizan para que los ingenieros, los interesados y el resto del equipo de desarrollo comprendan y acuerden los requerimientos. El arquitecto y/o los diseñadores del sistema reciben toda esta información y a partir de ella generan la arquitectura y el diseño del sistema. El cliente y la empresa desarrolladora pueden usarla para anexarla al contrato comercial. En particular los ejemplos pueden convertirse en el criterio de corrección y aceptación del sistema: si todos los ejemplos definidos durante la ingeniería de requerimientos son ejecutados sobre el sistema final y este da las respuestas por ellos mismos descriptas, entonces el sistema es correcto y el cliente debe aceptarlo. Claramente, los prototipos no se desechan sino que sirven hasta la entrega del sistema.

Sin embargo, son desechables porque no se programa sobre ellos. Es decir, los diseñadores no usan el diseño de los prototipos para generar el diseño del sistema, ni los programadores usan el código de los prototipos para seguir programando.

1.2. Repose: una herramienta para prototipación rápida desechable

Repose es un desarrollo experimental de Flowgate Security Consulting y personal de CIFASIS¹ que implementa la metodología descripta en la sección anterior. Una característica interesante de nuestra herramienta es que solo se necesita saber usar software de oficina más o menos estándar para poder aplicarla; el usuario de nuestra herramienta no necesita saber programar para desarrollar prototipos en muy poco tiempo.

En tanto es aun un desarrollo experimental se debe tener en cuenta que no se han programado las comprobaciones de entrada habituales por lo que los ingenieros deben ser sumamente cuidadosos al usar la herramienta, ya que de lo contrario el comportamiento será impredecible.

Lo que sigue es una suerte de manual de usuario de la herramienta.

¹CIFASIS es el Centro Franco-Argentino de Ciencias de la Información y de Sistemas, instituto de investigación que depende de CONICET, Universidad Nacional de Rosario y Universidad de Marsella.

1.2.1. Requisitos para instalar la herramienta

Repose es un programa Java que interactúa con OpenOffice.org. Entonces los requisitos para instalar Repose son los siguientes:

- Linux o Windows (XP, Vista, 2000, 2003, etc.)
- Java SE Runtime Environment 1.6 o superior
- OpenOffice.org 2.3.0 o superior (tener en cuenta que Repose usa la definición de OpenDocument Format de OpenOffice.org la cual suele variar de versión en versión por lo que podría dejar de funcionar en futuras versiones de OpenOffice.org)
- 7-Zip para Windows o gunzip para Linux

Para instalar Repose en Linux solo hay que descomprimir el archivo `repose.tar.gz` en cualquier directorio. Para hacerlo sobre Windows las instrucciones son las siguientes:

1. Descomprimir el archivo `repose.tar.gz` en cualquier carpeta.
2. Instalar 7-Zip, cuyo instalador puede descargarse gratuitamente desde <http://www.7-zip.org>.
3. Una vez instalado 7-Zip, buscar en la carpeta donde se lo instaló (la carpeta por defecto es `C:\Archivos de programa\7-Zip`) el archivo `7z.exe` y copiarlo en la carpeta donde se haya instalado Repose.

1.2.2. Panorama general de Repose

En primer lugar debemos aclarar que el ámbito de Repose es prototipar aplicaciones con rica interacción con el usuario a través de menús, formularios y otros objetos de interfaces gráficas de usuario (GUI).

El primer paso antes de usar Repose es dividir los requerimientos en función de formularios o pantallas a través de las cuales interactuará el usuario con la aplicación. Puede pensarse que detrás de cada formulario se implementan las reglas de negocio que corresponden al requerimiento. Por ejemplo, si el requerimiento es “procesar facturas”, entonces podemos dividirlo en “hacer una factura”, “borrar una factura” y “modificar una factura” y asociar a cada uno de ellos formularios que les permitirán a los usuarios realizar esas tareas.

El segundo paso es listar todos los datos que cada formulario necesita clasificándolos en datos de entrada, datos de estado y datos de salida. Los datos de entrada son los que ingresa el usuario (incluyen campos de texto, botones, listas desplegables, etc.), los datos de estado son aquellos que se toman de una base de datos o de un archivo (aunque no es necesario indicar ni la base de datos ni el archivo), y los datos de salida son los que produce el programa luego de ejecutar (por ejemplo un mensaje de error, un listado, etc.). Con toda esta información se define o especifica el prototipo como se describe en la sección 1.2.3; aquí solo daremos una descripción general.

Primero se usa el programa OpenOffice.org Draw para dibujar cada formulario como se muestra en la Figura 3. Cada figura geométrica representa un control de interfaz de usuario diferente; por ejemplo, los rectángulos con bordes redondeados representan botones. Luego, a partir del formulario, Repose genera el esqueleto de una planilla de cálculo donde el ingeniero debe definir el comportamiento del prototipo. Usando OpenOffice.org Calc el ingeniero debe listar los ejemplos que definirán el comportamiento del prototipo como se muestra en la Figura 4. Finalmente, se compila y ejecuta el prototipo. Cuando se ejecuta el prototipo se abren dos ventanas (Figura 5): una es la ventana propia del programa que se está prototipando, la otra es la ventana que muestra la base de datos, llamada *ventana de estado*. Observar que la ventana de aplicación tiene una barra de menú cuyo

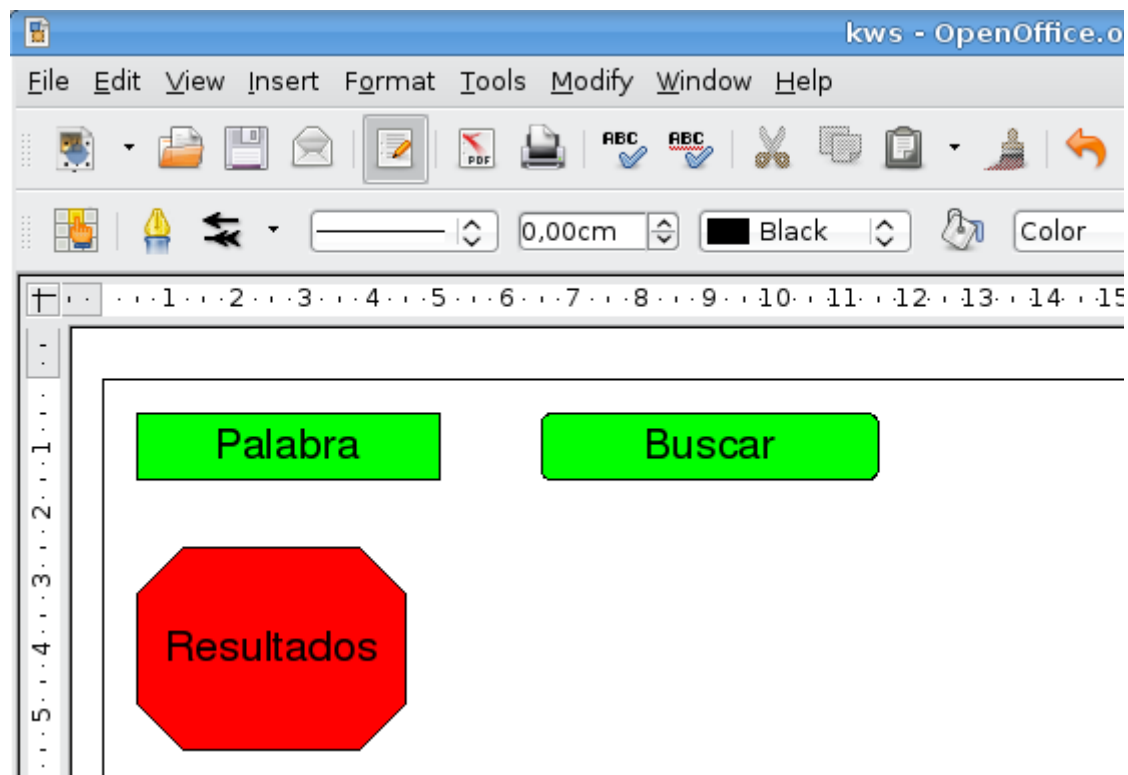


Figura 3: Dibujando la GUI.

contenido puede ser definido por el ingeniero. Los formularios que se hayan dibujado pueden activarse seleccionando el menú y la opción de menú apropiadas. Una vez que se elige la opción de menú, el formulario correspondiente se dibuja sobre el fondo de la ventana de aplicación (Figura 6) o como un cuadro de diálogo. Luego, el usuario puede ingresar datos, picar en los botones, seleccionar ítem de una lista desplegable, etc. Dependiendo de los ejemplos que se hayan definido en la planilla de cálculo y de la entrada del usuario, en algún momento el prototipo producirá alguna salida y/o algún cambio de estado (que se podrá apreciar en la ventana de estado).

1.2.3. Cómo usar Repose

En esta sección explicamos con detalle cómo desarrollar un prototipo². Cada prototipo se desarrolla en cinco pasos:

1. Modelar el prototipo
2. Dibujar la interfaz de usuario (GUI)
3. Definir el comportamiento del prototipo
4. Compilar el prototipo
5. Ejecutar el prototipo; es decir, permitir que los interesados jueguen con el prototipo

En los párrafos que siguen se explica cada uno de estos pasos.

²En todo lo que sigue se asume que su versión de OpenOffice.org está en castellano.

kws - OpenOffice.org Calc						
File Edit View Insert Format Tools Data Window Help						
Nimbus Sans L 10						
A1 f(x) Σ = Begin Prototype						
	A	B	C	D	E	F
1	Begin Prototype					
2	Form	Buscar (Herramientas)	kws.odg			
3	Requirement					
4	ID	Priority	Quality	Stakeholders		
5	F23	Impementar	Sin validar	Gustavo Deco		
6	Description	Mostrar los títulos que contienen la palabra clave				
7	Label	varin Palabra	varin Buscar	varout Resultados	varst Titulos	
8	Comenzar	Alto	Action	[Alto, El Alto]	[Alto, Bajo, El Alto]	
9	NoLabel	Bajo	Action	[Bajo]	[Alto, Bajo, El Alto]	
10	NoLabel	Medio	Error	[]	[Alto, Bajo, El Alto]	
11						
12	Form (NoMenu)	Error	error.odg			
13	Requirement					
14	ID	Priority	Quality	Stakeholders		
15	E10	Impementar	Validado	Ninguno en particular		
16	Description	Ventana general de error				
17	Label	varin OK				
18	Error	Comenzar				
19						
20	Init State					
21	Titulos					
22	[Alto, Bajo, El Alto]					
23	End Prototype					

Figura 4: Los ejemplos que definen el comportamiento del prototipo.

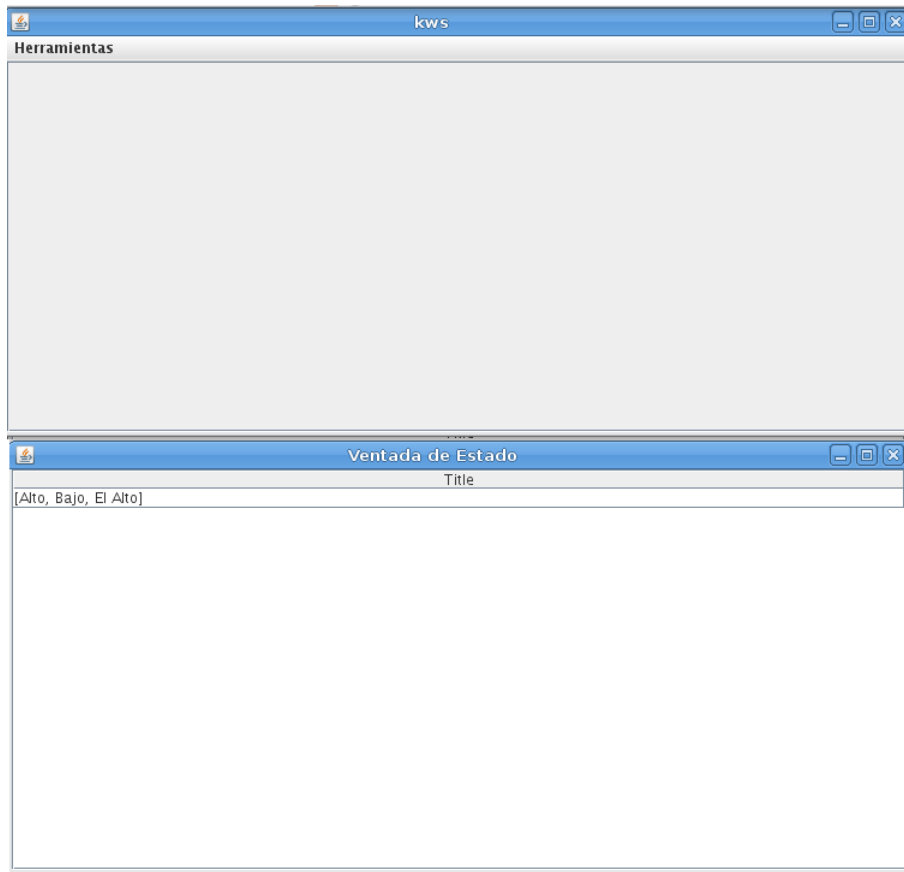


Figura 5: Las ventanas típicas de un prototipo. La ventana superior es la de la aplicación propiamente dicha; la ventana inferior es la que muestra los datos almacenados en la base de datos en cada momento.

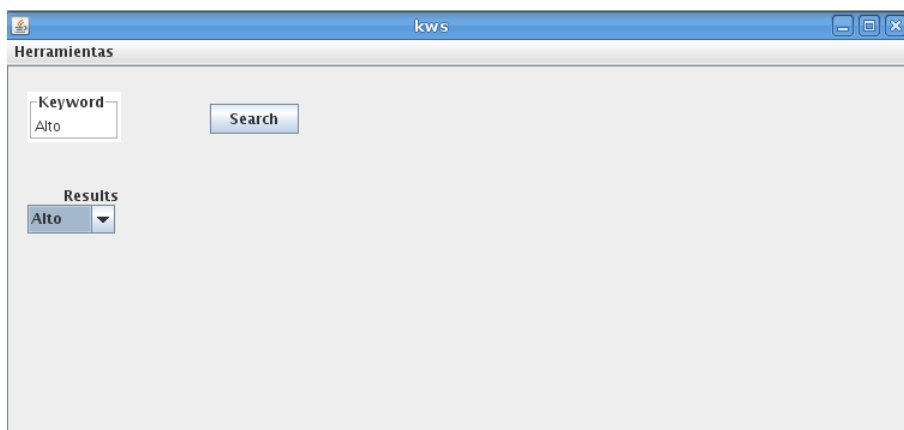


Figura 6: El usuario puede activar un formulario seleccionándolo en la barra de menús, y luego puede probar los ejemplos especificados en la planilla de cálculo.

Modelar el prototipo. Modelar un prototipo involucra:

- Pensar el prototipo en términos de formularios y pensar cada formulario en términos de datos de entrada, de estado y de salida.
- Definir los formularios del prototipo, lo cual a su vez involucra:
 - Definir los datos de entrada para ese formulario (nombre y tipo de control de interfaz).
 - Definir los datos de estado para ese prototipo (nombre). Los datos de estado son globales para todos los prototipos (es decir, todos los prototipos comparten los datos de estado).
 - Definir los datos de salida para ese formulario (nombre y tipo de control de interfaz).



Los ingenieros solo deben escribir esta información en algún lugar o simplemente tenerla en mente.

Dibujar la interfaz de usuario (GUI). Las GUIs de los prototipos se dibujan usando OpenOffice.org Draw. Asumimos que el lector es capaz de usar este tipo de programas. Al dibujar GUIs para los prototipos de Repose, deben tenerse en cuenta las siguientes notas.

- Crear un archivo odg³ por cada formulario del prototipo.

La página del archivo representa la pantalla de la computadora. Por lo tanto, el ingeniero debe distribuir en la página los controles de interfaz tal y como desea que se vean en la pantalla, teniendo en cuenta los puntos que siguen.

Cada control de interfaz corresponde a un dato de entrada, estado o salida.

- Llevar la página de cada archivo odg a orientación horizontal y poner los márgenes en cero.
- Los controles de interfaz se dibujan usando algunas de las *Formas Básicas* y las *Formas de Símbolos* de OpenOffice.org Draw. Estas figuras geométricas se obtienen pulsando los botones  y  de la barra de dibujo, como se muestra en la Figura 7.

Además, el fondo de cada forma geométrica debe llenarse en un color diferente dependiendo de si es un dato de entrada (Verde claro), estado (Amarillo) o salida (Rojo claro). Es importante seleccionar específicamente los colores con esos nombres pues de lo contrario Repose no funcionará correctamente.

La Tabla 1 muestra la relación entre figuras geométricas, controles de interfaz de usuario y colores actualmente soportados por Repose. Observar que no todos los controles pueden usarse para representar datos de entrada, estado o salida. Por ejemplo, los *checkbox* solo pueden usarse para datos de entrada.

Un dato es de entrada cuando lo provee el usuario; estos incluyen a los botones que pulsa el usuario.

Los datos de estado son aquellos que se almacenan en la base de datos. Por lo tanto una de las funciones principales de los formularios es convertir datos de entrada en datos de estado.

Un dato es de salida cuando es producido por el sistema.

Cuando en un formulario se incluye en Amarillo un dato de estado significa que el usuario podrá seleccionarlo como un dato de entrada. Esto facilita al ingeniero tomar datos que ya están en el sistema para que el usuario seleccione los que desea.

Las listas editables (es decir los octógonos) permiten que el usuario ingrese o elimine de a un valor por vez. Para ello se provee de forma automática de dos botones: Ins, para insertar

³odg es la extensión estándar de los archivos generados por OpenOffice.org Draw.

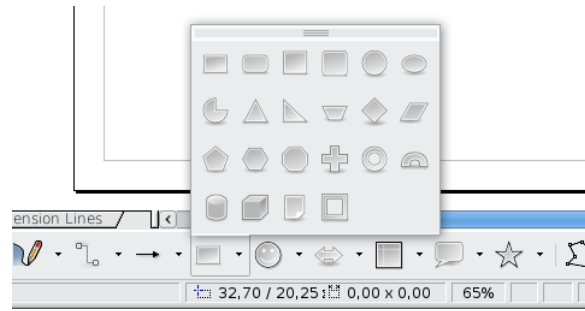


Figura 7: Para dibujar los controles de interfaz deben usarse las Formas Básicas y las Formas de Símbolos de la barra de dibujo. Aquí solo se muestran las primeras.


Control	Figura	Nombre y Grupo	Entrada	Estado	Salida
Caja de texto		Rectángulo Formas Básicas			
Botón		Rectángulo redondeado Formas Básicas			
Lista desplegable		Hexágono Formas Básicas			
Lista editable		Octógono Formas Básicas			
<i>Check box</i>		Papel Formas Básicas			
Cuadro de diálogo		Square Bevel Símbolos			

Tabla 1: Controles gráficos, colores y figuras geométricas. Los cuadros de diálogo se usan para organizar los otros controles de interfaz por lo que no son de entrada, de estado ni de salida. Los colores permitidos son: Verde claro, Amarillo y Rojo claro.

valores en la lista (en su lugar se puede pulsar Enter); y Sup, para eliminar de la lista el valor seleccionado. Por el contrario, las listas desplegables (es decir los hexágonos) no son editables: solo se pueden ver sus valores (cuando son datos de salida) o se puede seleccionar uno de ellos (cuando son datos de entrada o de estado).

- Dentro de cada figura geométrica escribir el nombre con que se quiere etiquetar cada control una vez que el prototipo esté en funcionamiento, como se muestra en la Figura 8.
- Cuando el prototipo está en ejecución, los formularios pueden mostrarse sobre el fondo de la ventana de aplicación o sobre un cuadro de diálogo. Para lograr el primer efecto simplemente se dibujan los controles de interfaz sobre la página del archivo odg. En cambio para lograr el segundo efecto se debe dibujar primero un Square Bevel sobre la página del archivo, y luego los controles de interfaz dentro de aquel, como se muestra en la figura 9.

Los Square Bevel pueden tener cualquier color. No se debe incluir una etiqueta para los Square Bevel.

- Se puede escribir texto libre dentro de cada formulario, como se puede ver en la Figura 10. Esto se logra utilizando la herramienta de OpenOffice.org para escribir cajas de texto (). Este texto no es dato y no es posible modificarlo durante el funcionamiento del prototipo. Es útil para escribir carteles de error, ayuda, comentarios aclaratorios, títulos, etc. El texto se dibuja siguiendo las mismas premisas que para los otros controles de interfaz.

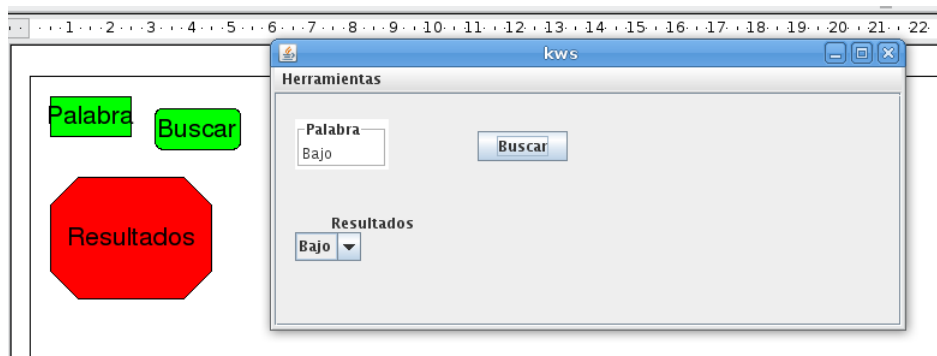


Figura 8: Los nombres dados a las figuras geométricas se usan más tarde para etiquetar el control de interfaz correspondiente.

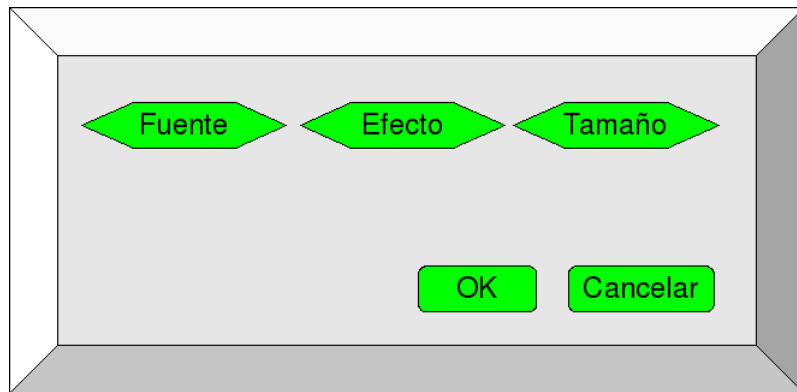


Figura 9: Para indicar que un formulario debe ser presentado como un cuadro de diálogo, y no en el fondo de la ventana de aplicación, se deben dibujar sus controles de interfaz dentro de un Square Bevel.

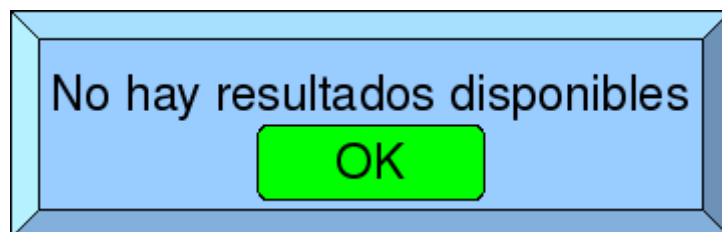


Figura 10: En este ejemplo se muestra el uso de cajas de texto para mostrar un cartel de error dentro de un Square Bevel, es decir de un cuadro de diálogo.

- Al ubicar cada forma en la hoja, tener en cuenta lo que sigue:
 - Repose no tiene en cuenta el tamaño de las formas básicas.
 - Repose tiene en cuenta el tamaño de los Square Bevels (que se usan para mostrar cuadros de diálogo).
 - Ubicar los dibujos cerca del ángulo superior izquierdo de la hoja, pues de lo contrario Repose generará un formulario con barras de desplazamiento.
 - Repose tiene en cuenta únicamente las coordenadas del ángulo superior izquierdo de cada forma básica.
 - El ancho de cada forma será el suficiente como para poder acomodar el valor más ancho que se use en los ejemplos para esa variable. Solo tendrá en cuenta las dimensiones de los cuadros de diálogo, es decir de los Square Bevels.
- En resumen, el usuario debe dibujar y nombrar una figura geométrica por cada dato de acuerdo al control de interfaz con que se quiera representar esa dato en el formulario.
- Guardar los archivos en formato odg en el mismo directorio donde se haya instalado Repose.

Definir el comportamiento del prototipo. El comportamiento del prototipo se define usando OpenOffice.org Calc. Se asume que el lector sabe cómo usar este tipo de aplicaciones. Al definir el comportamiento de un prototipo se deben tener en cuenta las siguientes notas.

- El comportamiento de todo el prototipo se especifica en una planilla de cálculo.
- El comportamiento del prototipo se define especificando el comportamiento asociado a cada formulario y cómo los formularios se relacionan entre sí (más precisamente cómo un formulario invoca a otros formularios).
- El comportamiento de cada formulario se especifica por medio de uno o más ejemplos de cómo se debe usar ese formulario.
- El esqueleto de esta planilla de cálculo puede generarse a partir de todos los archivos odg, aunque también puede crearse manualmente. Para generarla a partir de los archivos odg se debe usar el siguiente comando –solo podrá ejecutarlo desde el directorio de Repose.

```
java GenOds archivo.ods archivo_1.odg ... archivo_n.odg
```

Es decir que se debe indicar el nombre del archivo ods a crear a partir de los archivos odg que se pasan como parámetros. Cada archivo odg debe contener la definición gráfica de un formulario del prototipo.

Una vez que el archivo ods ha sido creado se debe usar OpenOffice.org Calc para modificarlo y completarlo como se indica más abajo.

Si no se usa el comando anterior se debe usar OpenOffice.org Calc para crear el archivo como se indica más abajo.

¡¡Importante!! Si se usa el comando anterior luego de que el archivo ods ha sido modificado por el ingeniero, se perderán todos los cambios. Es decir el comando crea un archivo desde cero sin considerar los datos que el usuario haya incorporado a la planilla anterior.

En lo que sigue se asume que el usuario va a crear el archivo ods manualmente. Como el comando **GenOds** ejecuta automáticamente varios de los pasos indicados más abajo, el lector puede obviarlos. De todas formas recomendamos leerlos a todos porque el archivo generado automáticamente no contiene todos los detalles. Más aun, recomendamos que la primera vez utilice el comando **GenOds** y complete los detalles que faltan manualmente.

¡¡Importante!! Recordar que Repose es aun un programa experimental por lo que no controla errores del usuario. Por lo tanto poner mucha atención en seguir los pasos al pié de la letra puesto que de lo contrario Repose no funcionará correctamente.

La planilla para describir el comportamiento de un prototipo se puede confeccionar en tres pasos:

1. Describir el comportamiento del primer formulario
2. Describir el comportamiento de más formularios
3. Describir los datos iniciales para los formularios

A continuación se detallan cada uno de estos pasos. Complemente lo que sigue leyendo la Figura 4.

Comportamiento del primer formulario.

1. Abrir una planilla de cálculo.
2. En la celda A1 escribir **Begin Prototype**.
3. En la celda A2 escribir **Form** o **Form (NoMenu)** (notar que entre **Form** y **(NoMenu)** hay un espacio en blanco). El primer caso se utiliza para los formularios que deben aparecer en algún menú de la barra de menús; el segundo se utiliza para los formularios que no deben aparecer en ningún menú, sino que son ejecutados desde otros formularios (por ejemplo una ventana que comunica un error).
4. En la celda B2 escribir el nombre de una opción de menú para uno de los formularios del prototipo y, dejando un espacio en blanco, el menú entre paréntesis. Por ejemplo **Guardar (Archivo)**. De esta forma cuando se ejecute el prototipo, el formulario se mostrará al seleccionar el nombre de menú y la opción indicadas. Si no se indican estos datos, al menos se debe indicar un nombre para el formulario. En ese caso el formulario se podrá ejecutar seleccionando el menú **Function** y la opción con el nombre del formulario. **GenOds** utiliza el nombre del archivo odg como nombre para el formulario.
5. En la celda C2 escribir el nombre del archivo odg que contiene la definición gráfica del formulario.
6. Las filas 3, 4, 5 y 6 se destinan para describir coloquialmente el requerimiento asociado al formulario. La Tabla 2 muestra la distribución de las celdas en esas filas.
Debajo de la columna **ID** se debe ingresar el identificador para el requerimiento; debajo de **Priority** la prioridad del requerimiento; debajo de **Quality** la calidad del requerimiento; y debajo de **Stakeholders** los interesados que participaron en la definición y validación del requerimiento.
En las celdas a la derecha de **Description** se debe ingresar la descripción coloquial del requerimiento. Este es tal vez el dato más importante de cada formulario.
7. En la celda A7 escribir **Label1**.

	A	B	C	D
3	Requirement			
4	ID	Priority	Quality	Stakeholders
5				
6	Description			

Tabla 2: Esta estructura debe repetirse para cada formulario de manera tal de poder describir cada requerimiento del sistema.

8. En la fila 7, comenzando por la columna B, y a razón de un dato por columna, escribir la definición de cada dato (de entrada, de estado y de salida) que usa el formulario. La definición de cada dato consta de:

clase debe ser uno de **varin**, para las variables de entrada; **varst**, para las variables de estado; y **varout**, para las variables de salida.

nombre debe ser uno de los nombres usados en el archivo odg o un nombre nuevo.

Se pueden agregar datos de estado pero no se pueden agregar datos de entrada o salida, respecto de los que están en el archivo odg. Todos los datos consignados en el archivo odg deben estar en la definición del formulario que se hace en la planilla.

Hay formularios que se utilizan para cambiar datos en la base de datos, es decir cambiar el estado del sistema. La forma que provee Repose para que el ingeniero indique un cambio en la base de datos es mediante *datos de estado primados*. Es muy simple: para cada dato de estado que sea modificado por el formulario se agrega una columna con el mismo nombre pero decorado al final con un apóstrofe. Por ejemplo, **usuarios** y **usuarios'**. La columna sin apóstrofe representa la base de datos antes de ejecutar el formulario y la columna con apóstrofe representa la base de datos luego de ejecutar el formulario. En resumen: se indica cómo está la base antes de ejecutar y cómo queda después de ejecutar, en dos columnas diferentes con el mismo nombre pero uno de ellos decorado con apóstrofe.

En aquellos formularios donde no se modifica la base de datos no es necesario definir las columnas decoradas con apóstrofe.

9. Repetir este paso cualquier número de veces, solo se debe incrementar el número de fila y tener en cuenta las indicaciones que se dan antes del paso siguiente. En la fila 8, comenzando por la columna B, escribir un valor en cada columna de manera tal que todos los valores de la fila representen una posible ejecución del formulario. Escribir *todos* los valores incluyendo los de salida ya que Repose no calculará nada por sí mismo.

A cada una de estas filas la llamamos *ejemplo*.

Tenga en cuenta que el tipo de los valores de un dato depende hasta cierto punto del control de interfaz que se eligió para representarlo. Las restricciones son las siguientes:

Botones. Los valores para los botones pueden ser **Action**, **NoAction** o el nombre de una **Label** de otro formulario –ver el punto 10.

Check boxes. Los valores pueden ser 0, que significa que el *check box* no se ha seleccionado; y 1, que significa que se ha seleccionado.

Listas desplegables. Los valores para las listas desplegables pueden ser cualesquiera pero siempre del mismo tipo (números o cadenas de caracteres). Si la lista desplegable es un dato de entrada, la lista de valores que se le muestre al usuario se armará con todos los

Label	varin TarjetasDeCredito	varout TarjetaElegida	varin Seleccionar	
NoLabel	Visa	Visa	Action	
NoLabel	Master	Master	Action	
NoLabel	Dinners	Dinners	Action	
NoLabel	American Express	American Express	Action	
End Prototype				

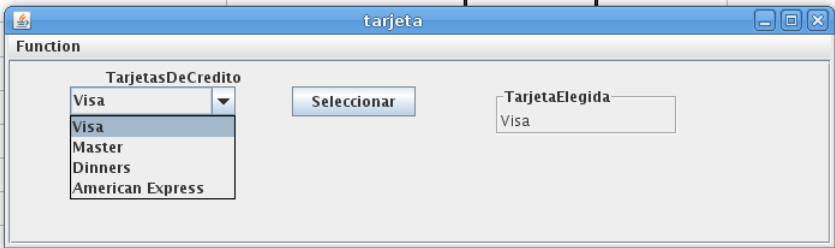


Figura 11: La lista TarjetasDeCredito es un dato de entrada por lo que la lista de valores se toma de los ejemplos.

valores distintos dados en la columna correspondiente del mismo formulario (Figura 11). De esta forma es simple armar la lista y a la vez especificar qué hay que hacer cada vez que se elige uno de los valores mostrados.

Listas. Si un dato se representa con un octógono (lista editable) los valores que se pueden ingresar en la columna correspondiente deben ser listas de valores. Es decir deben tener la forma `[val_1, . . . , val_n]`, donde cada `val` es una cadena de caracteres que no contiene una coma. También es posible indicar que la lista está vacía ingresando `[]`.

Otros valores. Para las otras columnas los valores pueden ser números, cadenas de caracteres o listas de valores, pero siempre deben ser del mismo tipo. Las listas de valores se encierran entre corchetes y cada valor se separa con una coma. Las listas de valores son buenas para mostrarlas en listas desplegables.

- La columna **Label** se utiliza para invocar un formulario desde otros formularios. Los formularios pueden invocarse al pulsar algún botón. Esto se consigue en dos pasos: (a) poniendo un nombre cualquiera en la columna **Label** del formulario al cual se quiere invocar, y (b) poniendo ese mismo nombre como **Action** de un botón del formulario desde el cual se quiere invocar al primero. Ver el ejemplo de la Figura 4 entre los formularios Buscar y Error.

Las celdas de la columna **Label** no pueden quedar en blanco. Por lo tanto, se debe escribir **NoLabel** si no se van a utilizar.

Uno de los usos más habituales para las etiquetas es poder mostrar ventanas de error, de ayuda, cuadros de diálogo más específicos, etc. que se invocan al pulsar diferentes botones en otros formularios.

Comportamiento de los otros formularios. Para especificar el comportamiento de más formularios se deben repetir los pasos anteriores excepto los dos primeros. Cada formulario se puede separar del anterior dejando una fila en blanco.

Datos iniciales para los formularios. Hemos visto que algunos formularios usan datos de estado que representan a la base de datos del sistema. Repose exige que se dé un valor inicial para cada columna que representa a la base de datos. Estos valores iniciales se establecen en una sección que se introduce dejando una fila en blanco luego del último ejemplo del último formulario. Digamos que la fila en cuestión es la N , entonces:

1. En la celda AN escribir **Init State**.
2. En la fila $N + 1$, comenzando en la columna A, y a razón de un dato de *estado* por columna, escribir el nombre del dato.
3. En la fila $N + 2$, escribir el valor inicial para cada dato.
4. En la celda $AN + 3$ escribir **End Prototype**.

Esta sección debe escribirse siempre a menos que en ningún formulario se haya utilizado variables de estado.

Además de las variables de estado, en esta sección de la planilla pueden darse más valores para las variables de entrada que se representan como listas desplegables. En alguna columna de la fila $N + 1$ se escribe el nombre de la variable y en la celda inferior se escriben los valores adicionales para mostrar al usuario, encerrados entre corchetes y separados por comas. Estos valores se adicionan a los valores que se hayan encontrado en el formulario donde se usa la variable.

¡¡Importante!! No olvide guardar el archivo en el mismo directorio donde está instalado Repose.

1.2.4. Compilar, ejecutar y modificar el prototipo

Una vez que se ha terminado de especificar el comportamiento del prototipo se debe abrir una ventana de línea de comandos, ir al directorio donde se haya instalado Repose y ejecutar los siguientes comandos:

1. `java ProtoGen archivo.ods archivo.java`

Genera el prototipo a partir de la planilla `archivo.ods` que es donde se ha especificado el comportamiento del prototipo, en tanto que `archivo.java` es el nombre que el usuario le debe dar al prototipo.

El comando escribe diversos mensajes en la pantalla. Si el último mensaje es **The prototype has been generated. Compile it with: javac archivo.java**, significa que el prototipo se ha generado con éxito; en caso contrario hay algún error en la planilla o en los archivos `odg` –por el momento Repose es muy poco informativo al respecto por lo que sugerimos mirar con atención la planilla tratando de descubrir algún error; los mensajes en la pantalla pueden ayudar.

2. `javac archivo.java`

Si el prototipo se ha generado con éxito el comando anterior crea un programa Java ejecutable. Si el prototipo ha sido generado con éxito el comando anterior no debería producir ningún error –si lo hace significa que Repose tiene algún error.

3. `java archivo`

Esta es la forma en que el prototipo debe ser ejecutado para que el cliente pueda jugar con él.

Una vez que el prototipo esta ejecutando se lo puede usar como a cualquier otro programa pero siempre se debe recordar que los únicos valores de entrada que aceptará son lo que están definidos en en los ejemplos. Además, recuerde que junto a la ventana de aplicación se abre también una ventana que muestra el estado actual de la base de datos a medida que se usa el prototipo.

Si hay que modificar el prototipo:

- Cerrar la ventana de aplicación –la ventana de la base de datos se cerrará automáticamente.

- Modificar cualquiera de los archivos `odg` agregando o borrando variables, o la representación gráfica de cualquiera de ellas.
- También se pueden crear otros archivos `odg` para agregar más formularios al prototipo.
- Modificar el archivo `ods` agregando, borrando o modificando ejemplos y/o agregando o borrando formularios (es decir, secciones **Form**).
- Volver a ejecutar los tres comandos anteriores en el mismo orden.

Si solo se quiere ejecutar el prototipo sin ninguna modificación (por ejemplo para mostrárselo a otro interesado) se debe ejecutar el último de los comandos anteriores.

Recuerde que el comando `java GenOds archivo.ods archivo_1.odg ... archivo_n.odg` le permite generar el esqueleto de la planilla donde especificar el comportamiento del prototipo cuyos formularios se especifican en los archivos `odg` `archivo_1.odg ... archivo_n.odg`. También recuerde que si la planilla ya fue generada la nueva planilla sobrescribirá a la anterior y usted perderá cualquier modificación que haya hecho.

Sea lo que fuere que se haga con *Repose* no hay que olvidar que el objetivo principal es

Obtener una lista consistente, completa y
acordada con el cliente de requerimientos funcionales

entonces siempre se debe poner mucho esfuerzo y cuidado en

Escribir el requerimiento asociado a cada formulario
y mantenerlo consistente con los ejemplos

porque es la información clave de todo el proceso.

Referencias

[Som95] Ian Sommerville. *Software engineering (5th ed.)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995.